**Assignment Number 07**

**Name:** Mihir Unmesh Patil

**Roll NO**: TYCOC213

**Batch:** C/C-3

**CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

int fcfs(int reference\_string[], int n, int capacity) {

    int frames[capacity];

    int frame\_index = 0;

    int page\_faults = 0;

    bool present;

    for (int i = 0; i < capacity; i++) {

        frames[i] = -1;

    }

    printf("FCFS Page Replacement:\n");

    for (int i = 0; i < n; i++) {

        present = false;

        for (int j = 0; j < capacity; j++) {

            if (frames[j] == reference\_string[i]) {

                present = true;

                break;

            }

        }

        if (!present) {

            // Page fault occurred

            page\_faults++;

            frames[frame\_index] = reference\_string[i];

            frame\_index = (frame\_index + 1) % capacity; // Circular queue

        }

        // Print current state of frames

        printf("Reference: %d | Frames: ", reference\_string[i]);

        for (int j = 0; j < capacity; j++) {

            printf("%d ", frames[j]);

        }

        printf("| Page Fault: %s\n", present ? "No" : "Yes");

    }

    return page\_faults;

}

// Function to implement LRU page replacement algorithm

int lru(int reference\_string[], int n, int capacity) {

    int frames[capacity];

    int time[capacity]; // To track the last time each page was used

    int page\_faults = 0;

    bool present;

    for (int i = 0; i < capacity; i++) {

        frames[i] = -1;

        time[i] = 0;

    }

    int current\_time = 0; // Global timer

    printf("LRU Page Replacement:\n");

    for (int i = 0; i < n; i++) {

        present = false;

        // Check if page is already in frame

        for (int j = 0; j < capacity; j++) {

            if (frames[j] == reference\_string[i]) {

                present = true;

                time[j] = ++current\_time; // Update last used time

                break;

            }

        }

        if (!present) {

            page\_faults++;

            int lru\_index = 0;

            for (int j = 1; j < capacity; j++) {

                if (time[j] < time[lru\_index]) {

                    lru\_index = j;

                }

            }

            frames[lru\_index] = reference\_string[i];

            time[lru\_index] = ++current\_time; // Update last used time

        }

        printf("Reference: %d | Frames: ", reference\_string[i]);

        for (int j = 0; j < capacity; j++) {

            printf("%d ", frames[j]);

        }

        printf("| Page Fault: %s\n", present ? "No" : "Yes");

    }

    return page\_faults;

}

int optimal(int reference\_string[], int n, int capacity) {

    int frames[capacity];

    int page\_faults = 0;

    bool present;

    for (int i = 0; i < capacity; i++) {

        frames[i] = -1;

    }

    printf("Optimal Page Replacement:\n");

    for (int i = 0; i < n; i++) {

        present = false;

        for (int j = 0; j < capacity; j++) {

            if (frames[j] == reference\_string[i]) {

                present = true;

                break;

            }

        }

        if (!present) {

            page\_faults++;

            int farthest = -1;

            int replace\_index = -1;

            for (int j = 0; j < capacity; j++) {

                int k;

                for (k = i + 1; k < n; k++) {

                    if (frames[j] == reference\_string[k]) {

                        if (k > farthest) {

                            farthest = k;

                            replace\_index = j;

                        }

                        break;

                    }

                }

                if (k == n) {

                    replace\_index = j;

                    break;

                }

            }

            if (replace\_index == -1) {

                for (int j = 0; j < capacity; j++) {

                    if (frames[j] == -1) {

                        replace\_index = j;

                        break;

                    }

                }

                if (replace\_index == -1) replace\_index = 0;

            }

            frames[replace\_index] = reference\_string[i];

        }

        printf("Reference: %d | Frames: ", reference\_string[i]);

        for (int j = 0; j < capacity; j++) {

            printf("%d ", frames[j]);

        }

        printf("| Page Fault: %s\n", present ? "No" : "Yes");

    }

    return page\_faults;

}

int main() {

    int reference\_string[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3};

    int n = sizeof(reference\_string) / sizeof(reference\_string[0]);

    int capacity = 3;

    int fcfs\_faults = fcfs(reference\_string, n, capacity);

    printf("\nFCFS Total Page Faults: %d\n", fcfs\_faults);

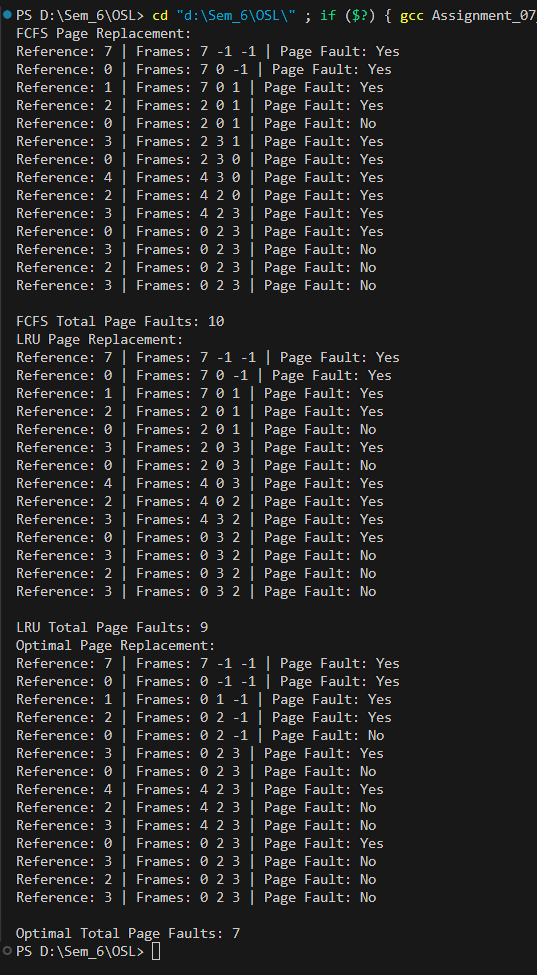
    int lru\_faults = lru(reference\_string, n, capacity);

    printf("\nLRU Total Page Faults: %d\n", lru\_faults);

    int optimal\_faults = optimal(reference\_string, n, capacity);

    printf("\nOptimal Total Page Faults: %d\n", optimal\_faults);

    return 0;

}

OUTPUT: